This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

OIPE JC56
MAY 27 2004
PATENT & TRADEMARK OFFICE

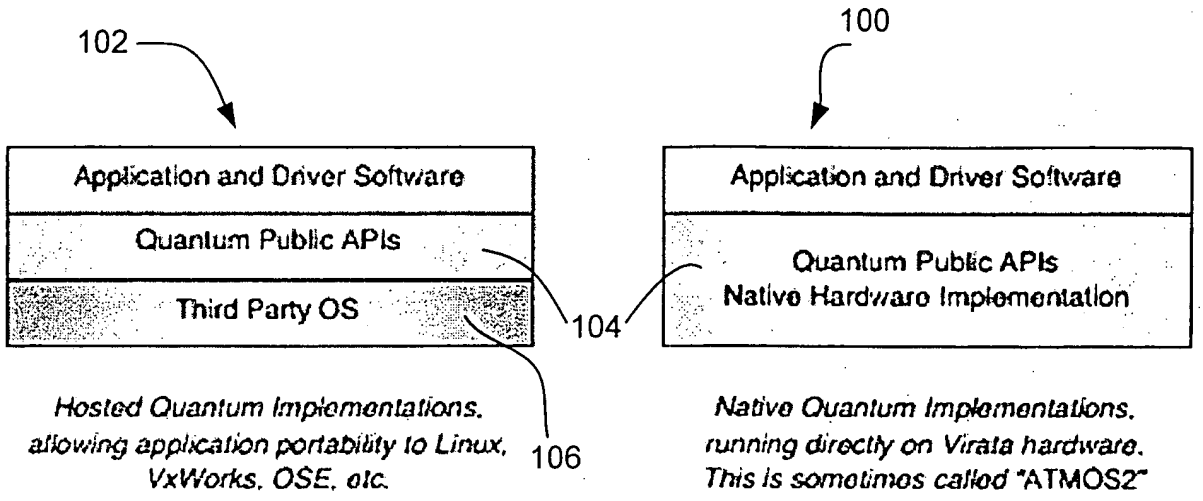

FIG. 1

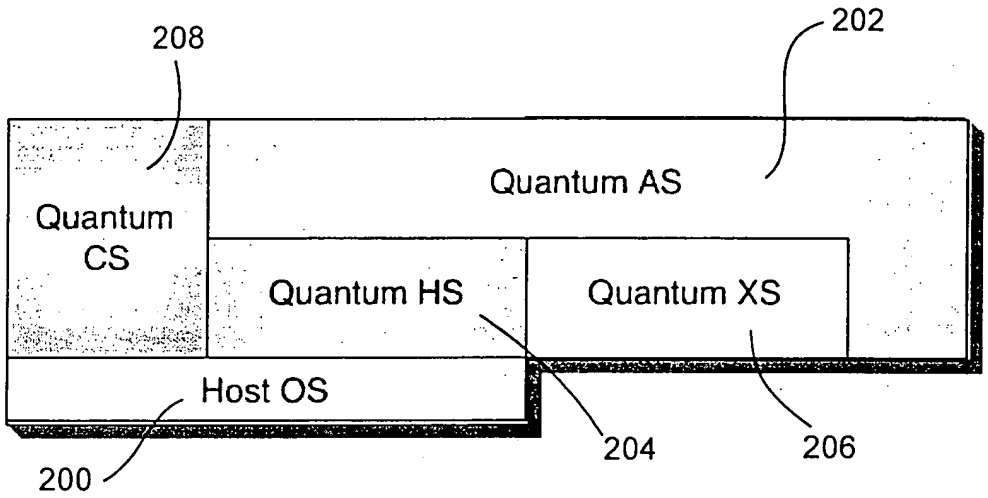

FIG. 2

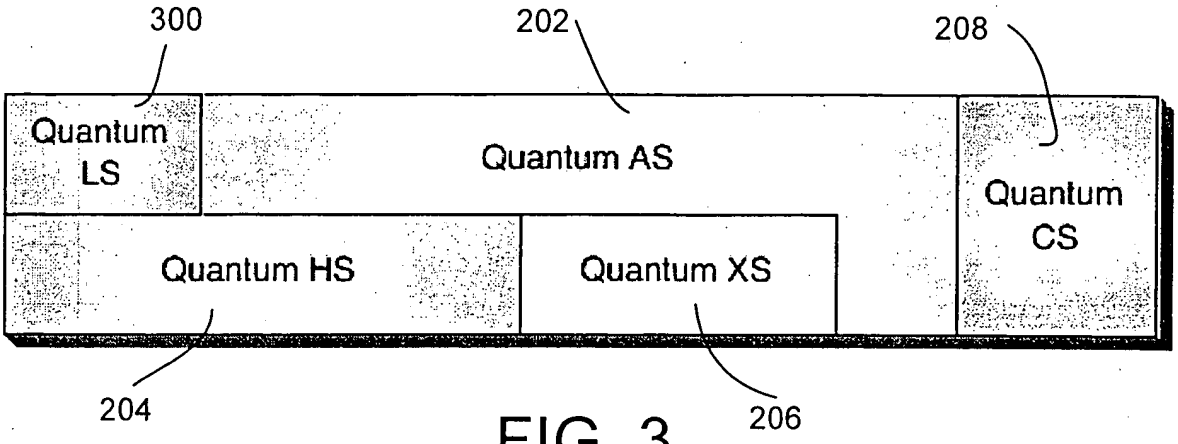
300
202
208
Quantum LS
Quantum AS
Quantum CS
Quantum HS
Quantum XS
204
206

FIG. 3

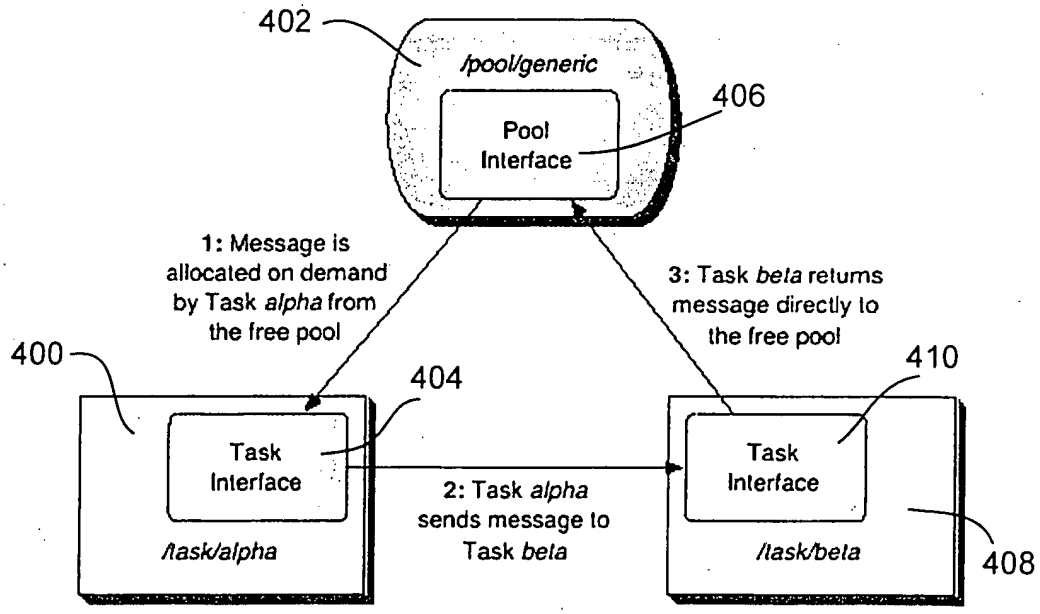
402
/pool/generic
406
Pool Interface
1: Message is allocated on demand by Task alpha from the free pool
3: Task beta returns message directly to the free pool
400
404
410
Task Interface
Task Interface
2: Task alpha sends message to Task beta
/task/alpha
/task/beta
408

FIG. 4

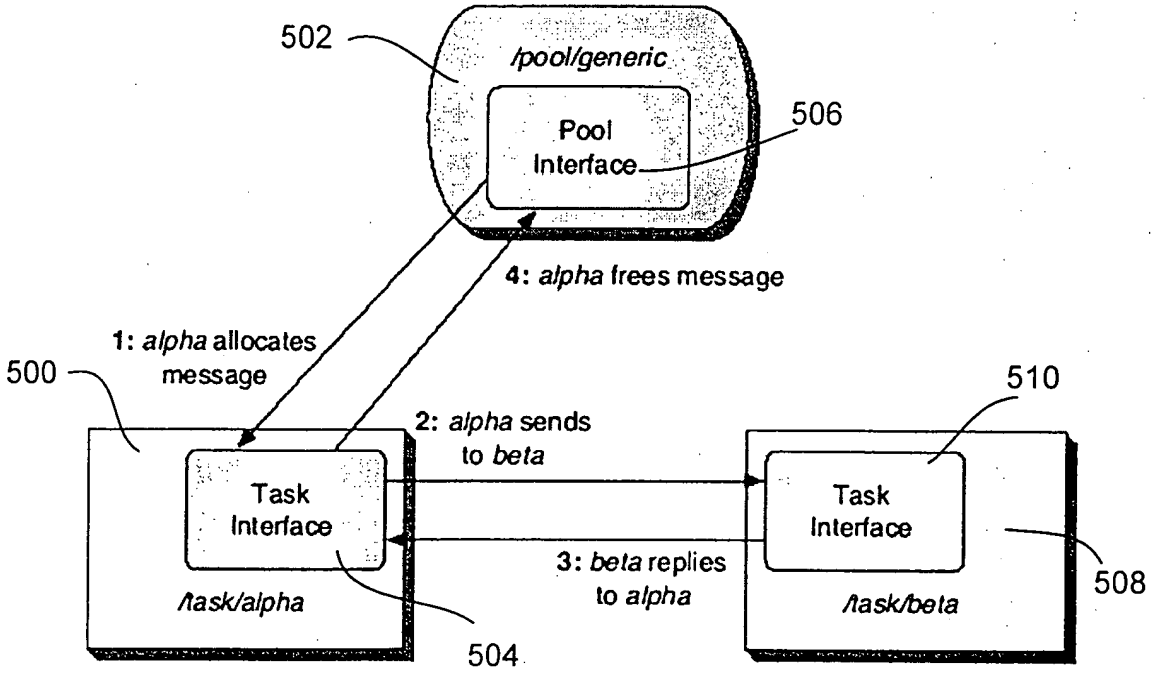
502
/pool/generic
Pool Interface
506
4: alpha frees message
1: alpha allocates message
500
2: alpha sends to beta
510
Task Interface
Task Interface
3: beta replies to alpha
508
/task/alpha
/task/beta
504

FIG. 5

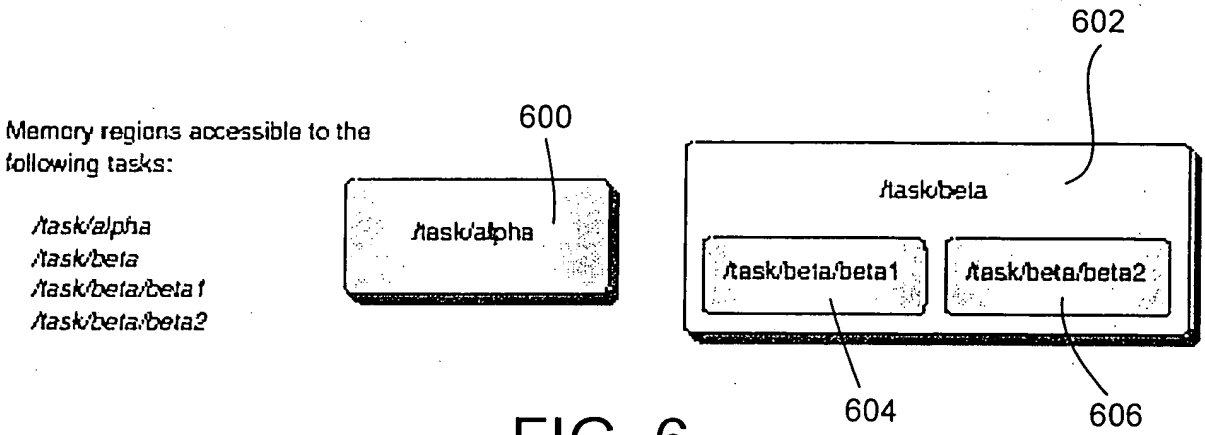
Memory regions accessible to the following tasks:
/task/alpha
/task/beta
/task/beta/beta1
/task/beta/beta2
600
/task/alpha
602
/task/beta
/task/beta/beta1
/task/beta/beta2
604
606

FIG. 6

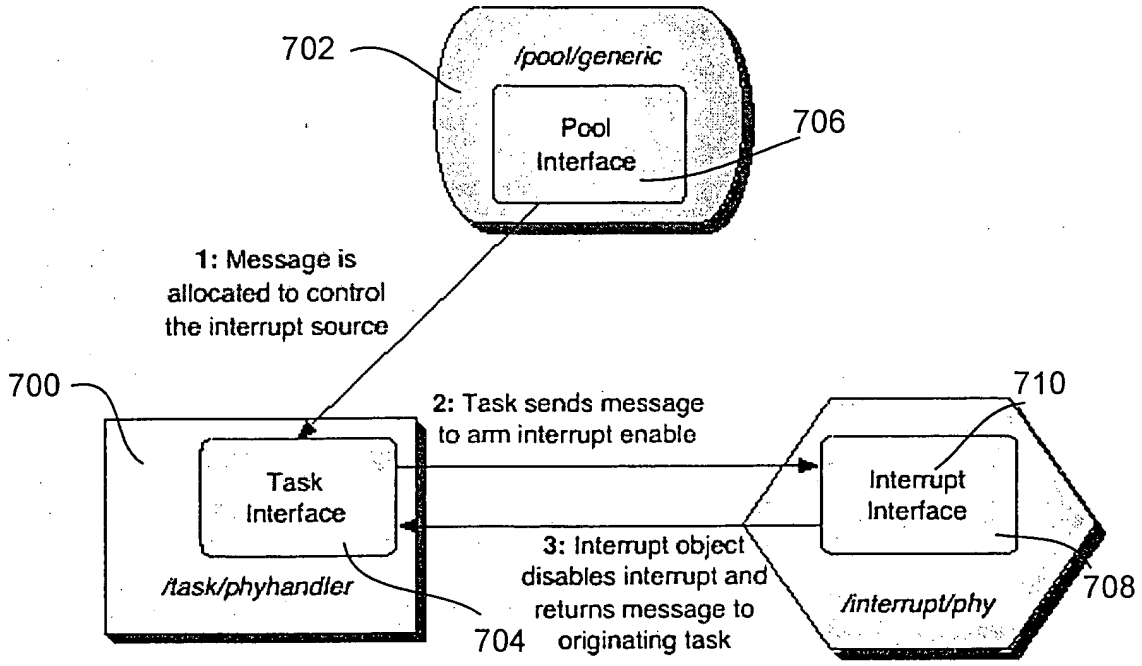
702
/pool/generic
Pool Interface
706
1: Message is allocated to control the interrupt source
700
2: Task sends message to arm interrupt enable
710
Task Interface
Interrupt Interface
3: Interrupt object disables interrupt and returns message to originating task
/task/phyhandler
/interrupt/phy
708
704

FIG. 7

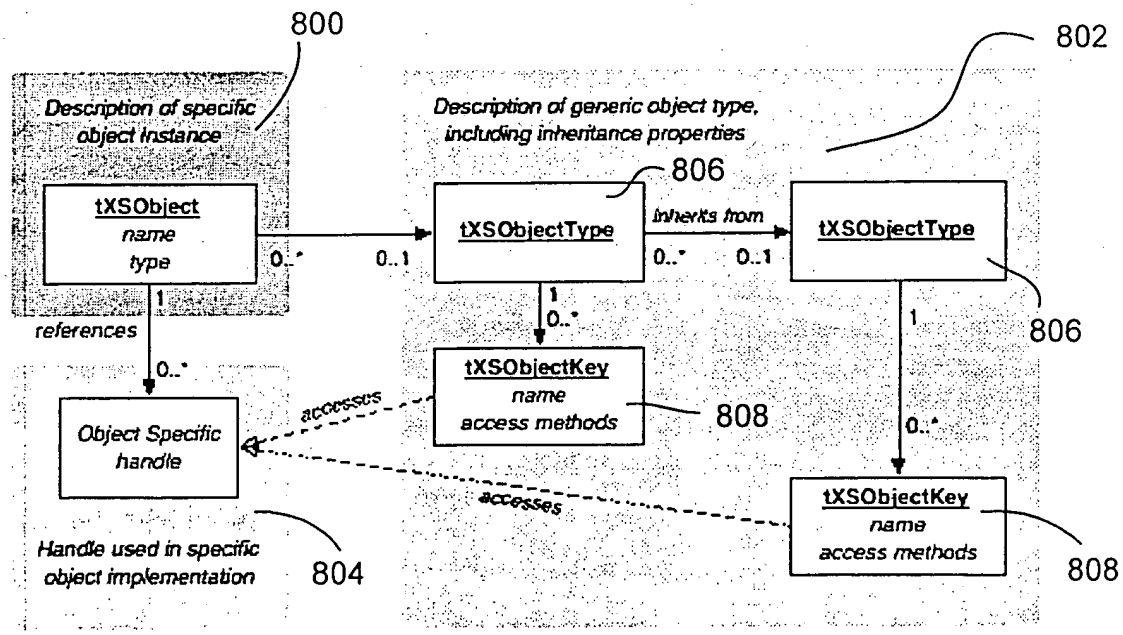
800
802
Description of specific object instance
Description of generic object type, including inheritance properties
806
tXSObject
name
type
tXSObjectType
Inherits from
tXSObjectType
0..*
0..1
0..*
0..1
1
0..*
1
0..*
806
references
tXSObjectKey
name
access methods
808
Object Specific handle
accesses
accesses
tXSObjectKey
name
access methods
808
Handle used in specific object implementation
804

FIG. 8

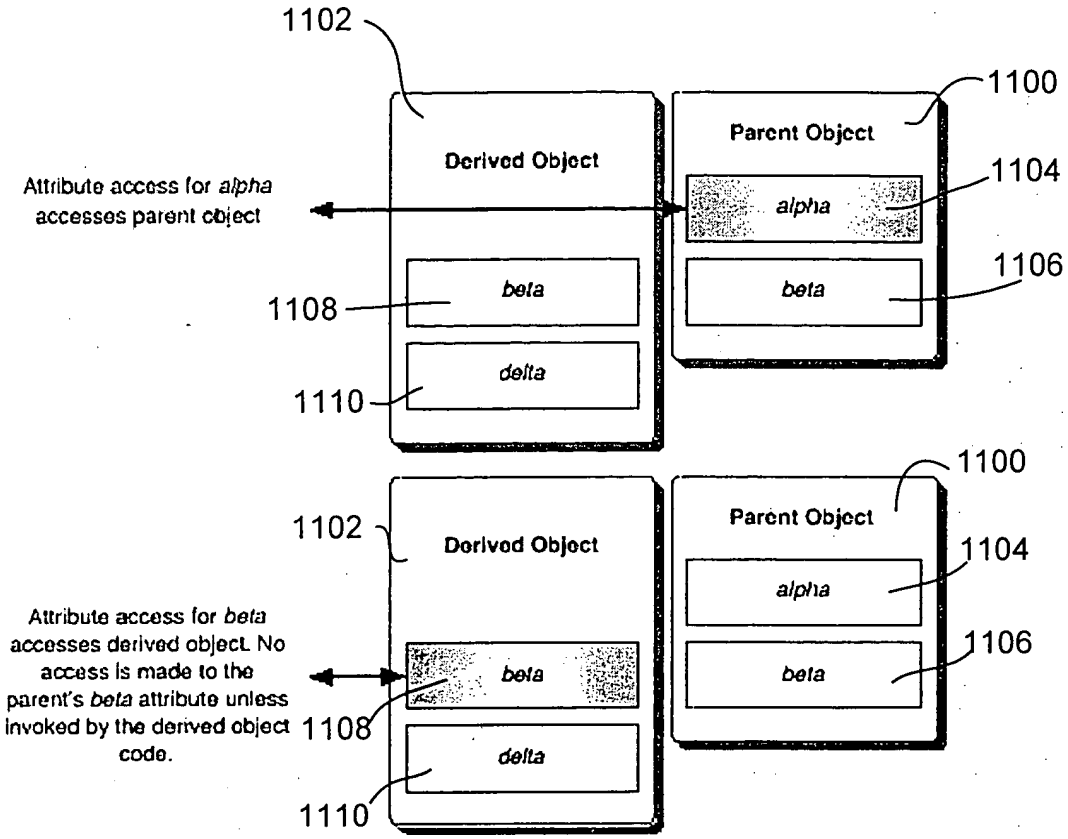
1102
Derived Object
1100
Parent Object
Attribute access for alpha accesses parent object
alpha
1104
beta
1106
1108
beta
delta
1110
1100
1102
Derived Object
Parent Object
1104
alpha
Attribute access for beta accesses derived object. No access is made to the parent's beta attribute unless invoked by the derived object code.
beta
1106
1108
beta
delta
1110

FIG. 11

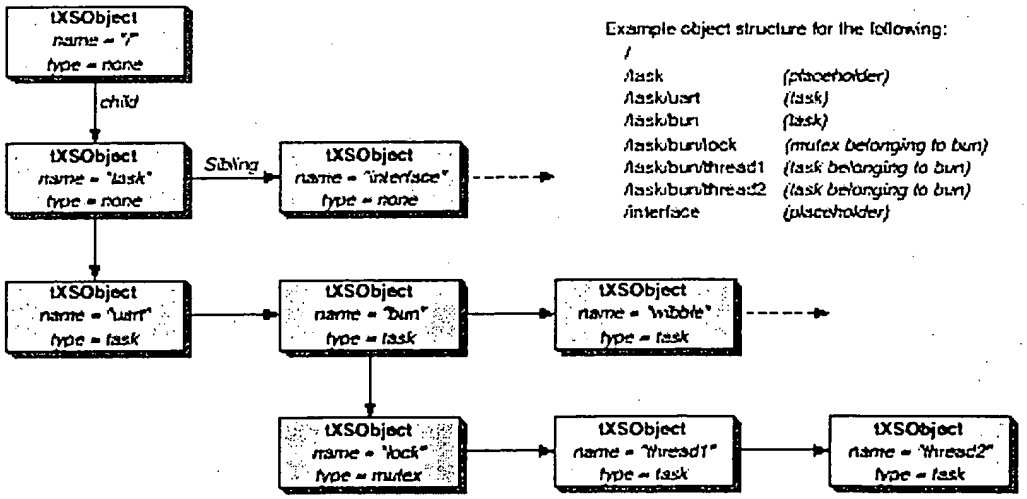
tXSObject
name = "/"
type = none
child
tXSObject
name = "task"
type = none
Sibling
tXSObject
name = "interface"
type = none
tXSObject
name = "uart"
type = task
tXSObject
name = "bun"
type = task
tXSObject
name = "wibble"
type = task
tXSObject
name = "lock"
type = mutex
tXSObject
name = "thread1"
type = task
tXSObject
name = "thread2"
type = task
Example object structure for the following:
/
/task (placeholder)
/task/uart (task)
/task/bun (task)
/task/bun/lock (mutex belonging to bun)
/task/bun/thread1 (task belonging to bun)
/task/bun/thread2 (task belonging to bun)
/interface (placeholder)

FIG. 12

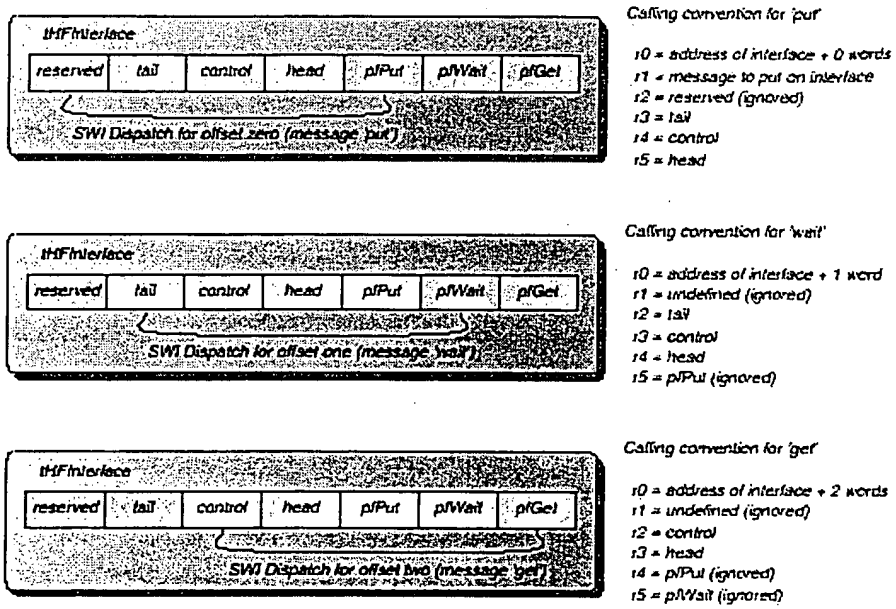
tHFInterface
reserved
tail
control
head
pfPut
pfWait
pfGet
SWI Dispatch for offset zero (message 'put')
Calling convention for 'put'
r0 = address of interface + 0 words
r1 = message to put on interface
r2 = reserved (ignored)
r3 = tail
r4 = control
r5 = head
SWI Dispatch for offset one (message 'wait')
Calling convention for 'wait'
r0 = address of interface + 1 word
r1 = undefined (ignored)
r2 = tail
r3 = control
r4 = head
r5 = pfPut (ignored)
SWI Dispatch for offset two (message 'get')
Calling convention for 'get'
r0 = address of interface + 2 words
r1 = undefined (ignored)
r2 = control
r3 = head
r4 = pfPut (ignored)
r5 = pfWait (ignored)

FIG. 13

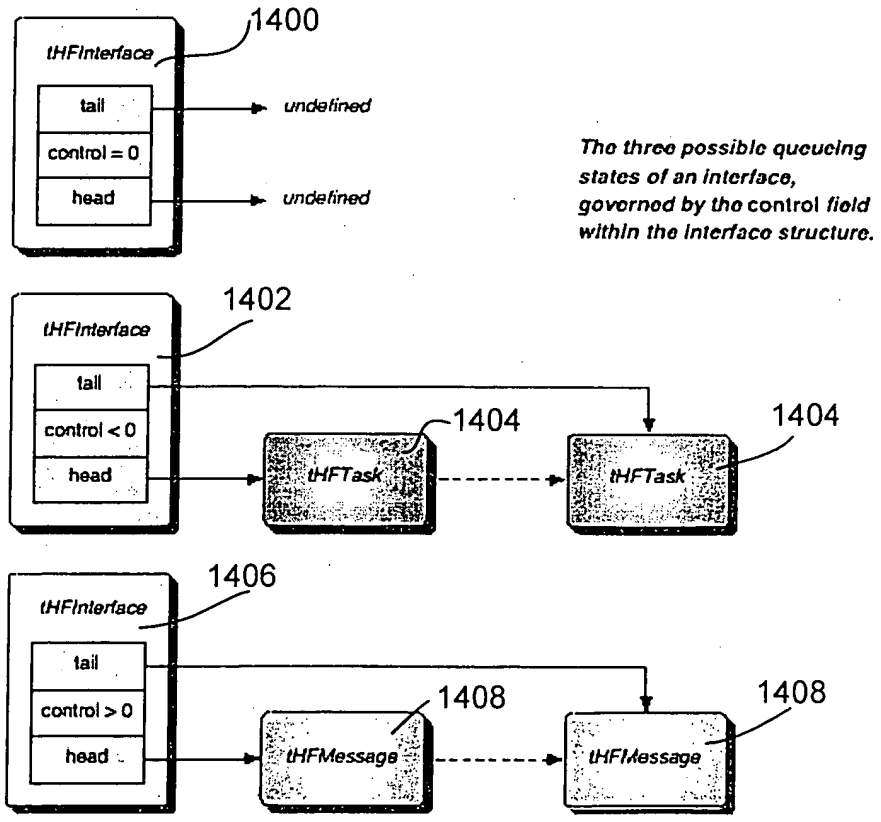
1400
tHFInterface
tail
control = 0
head
undefined
undefined
The three possible queueing states of an interface, governed by the control field within the interface structure.
1402
tHFInterface
tail
control < 0
head
1404
tHFTask
1404
tHFTask
1406
tHFInterface
tail
control > 0
head
1408
tHFMessage
1408
tHFMessage

FIG. 14

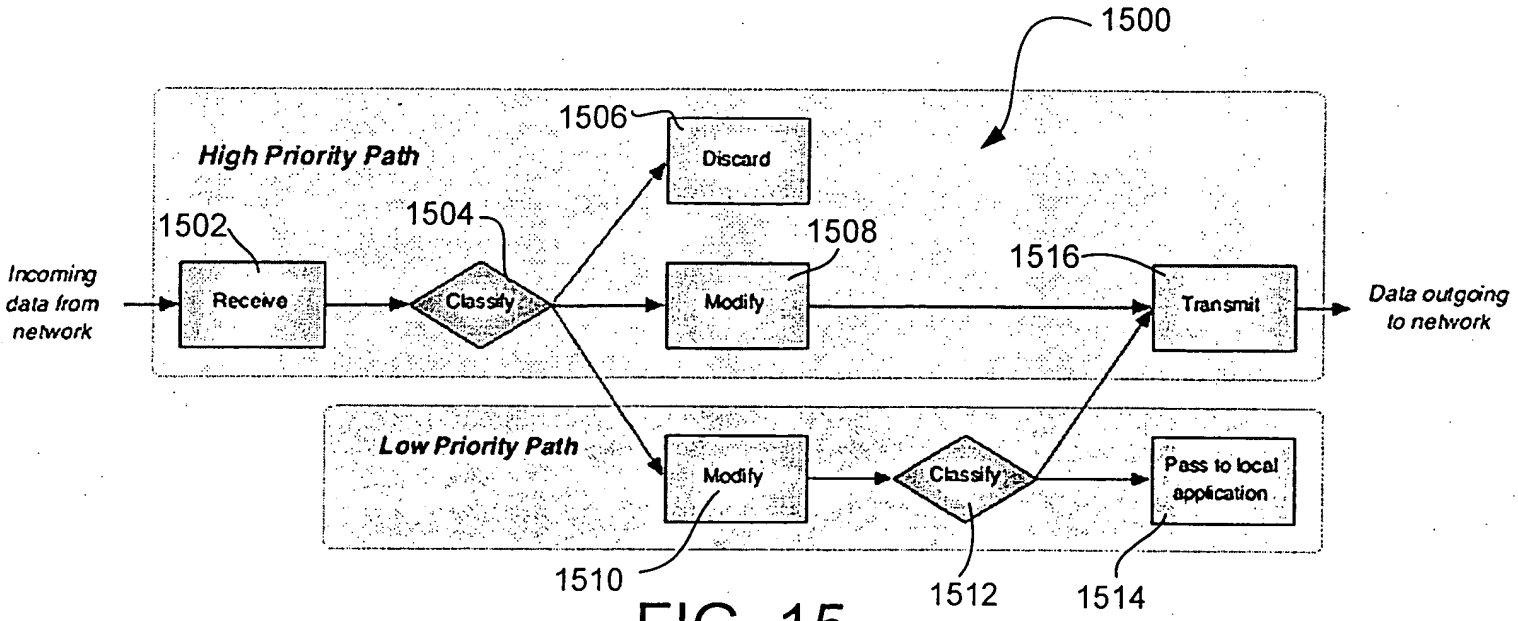
1500
High Priority Path
1506
Discard
1502
1504
1508
1516
Incoming data from network
Receive
Classify
Modify
Transmit
Data outgoing to network
Low Priority Path
Modify
Classify
Pass to local application
1510
1512
1514

FIG. 15

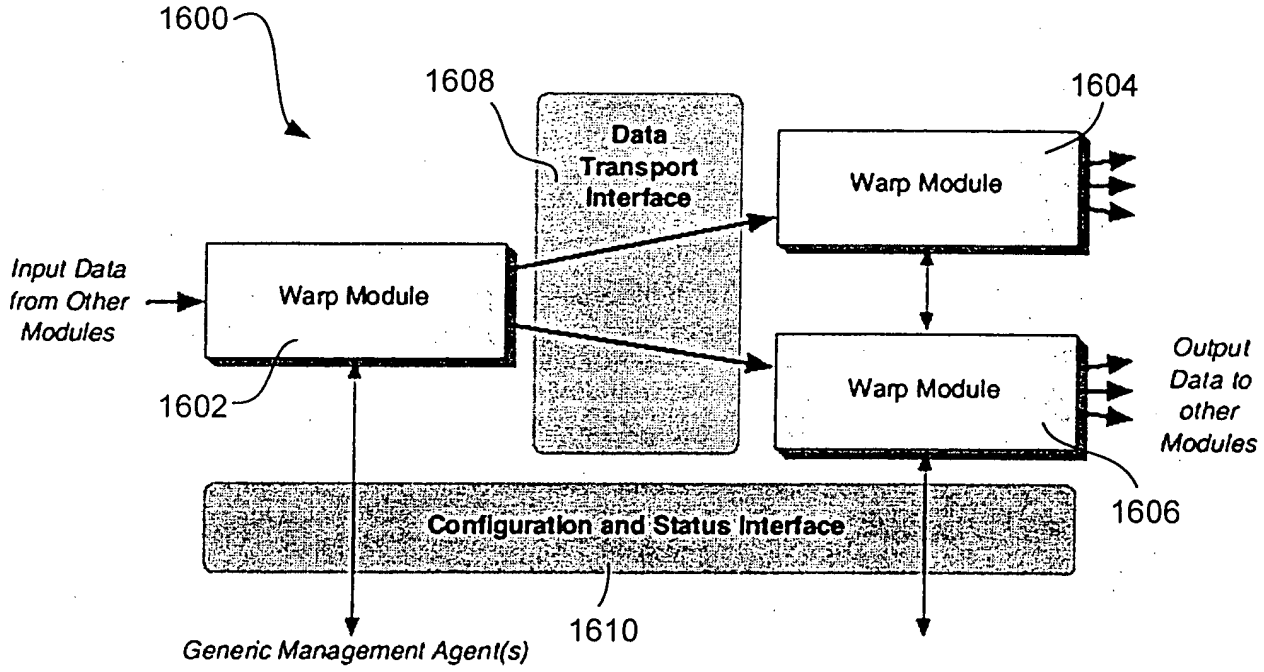
1600
1608
Data Transport Interface
1604
Warp Module
Input Data from Other Modules
Warp Module
Warp Module
Output Data to other Modules
1602
1606
Configuration and Status Interface
1610
Generic Management Agent(s)

FIG. 16

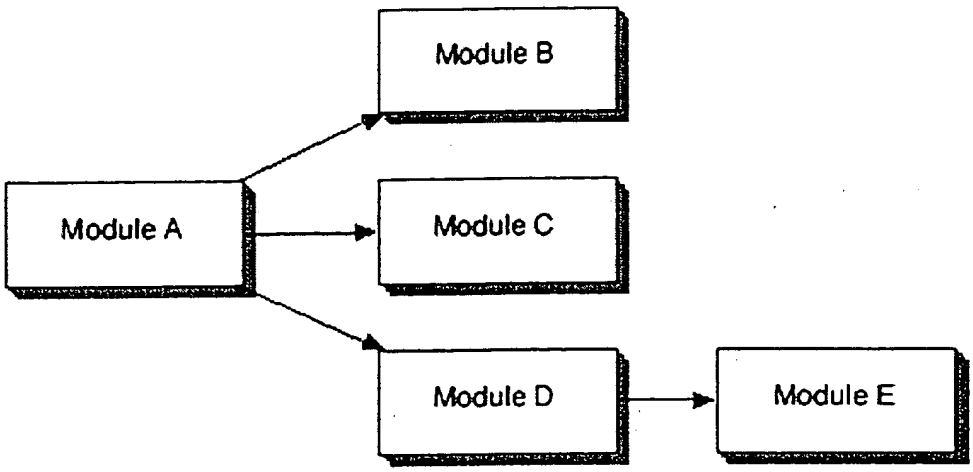
Module B
Module A
Module C
Module D
Module E

FIG. 17

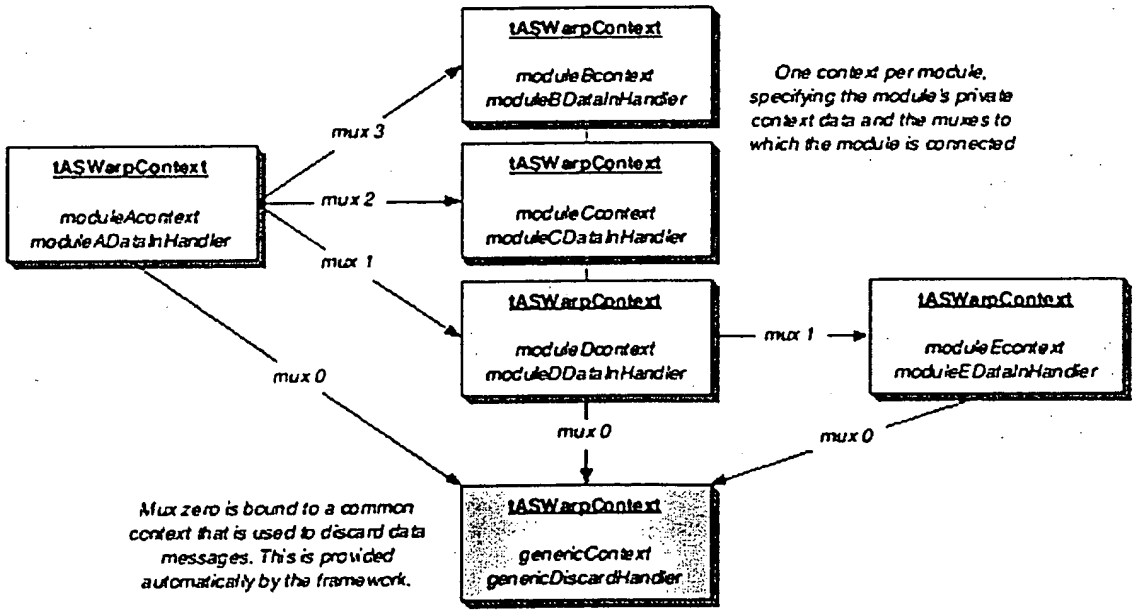
tASWarpContext
moduleBcontext
moduleBDataInHandler
One context per module, specifying the module's private context data and the muxes to which the module is connected
mux 3
tASWarpContext
moduleAcontext
moduleADataInHandler
mux 2
tASWarpContext
moduleCcontext
moduleCDataInHandler
mux 1
tASWarpContext
moduleDcontext
moduleDDataInHandler
mux 1
tASWarpContext
moduleEcontext
moduleEDataInHandler
mux 0
mux 0
mux 0
Mux zero is bound to a common context that is used to discard data messages. This is provided automatically by the framework.
tASWarpContext
genericContext
genericDiscardHandler

FIG. 18

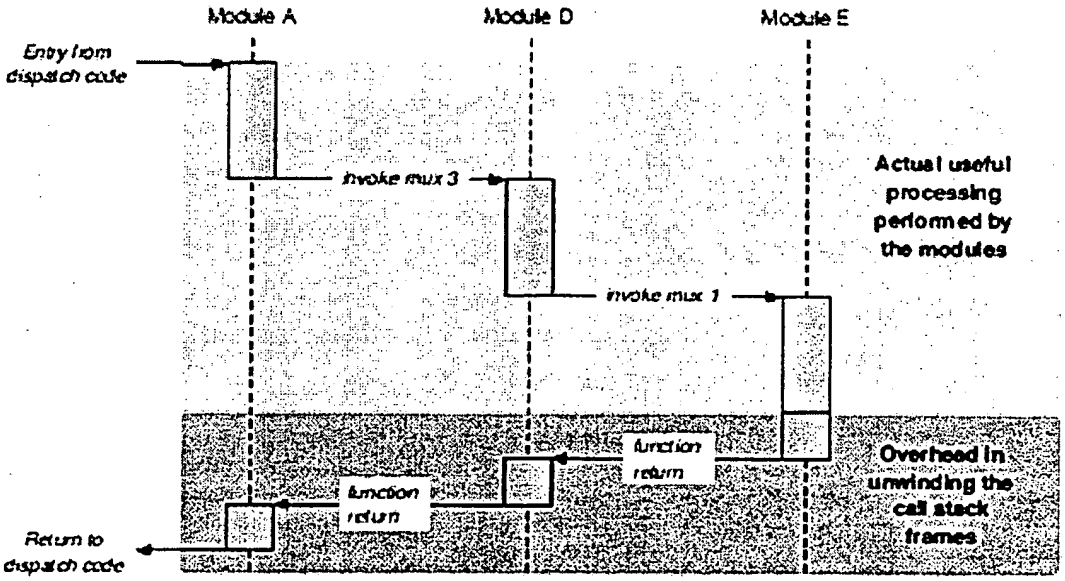
Module A
Module D
Module E
Entry from dispatch code
invoke mux 3
invoke mux 1
Actual useful processing performed by the modules
function return
function return
Overhead in unwinding the call stack frames
Return to dispatch code

FIG. 19

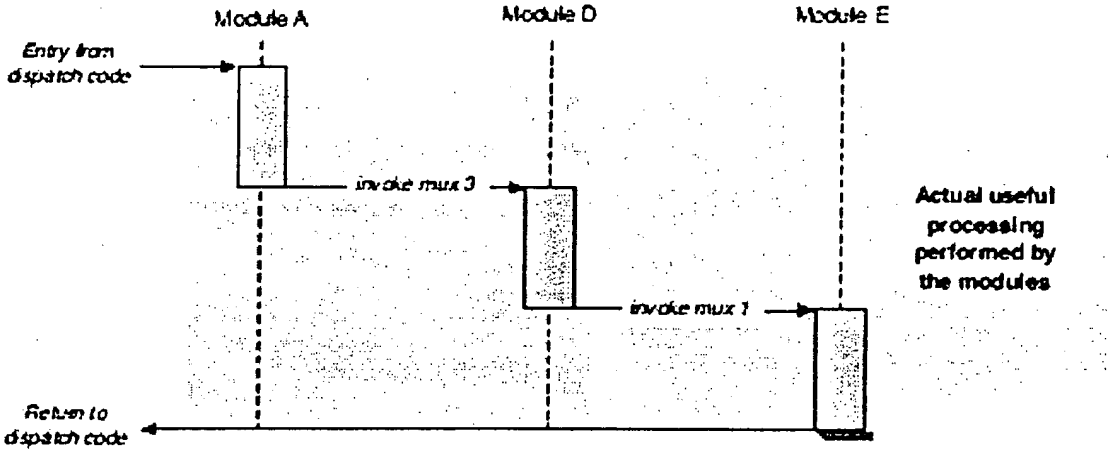
Module A
Module D
Module E
Entry from dispatch code
invoke mux 3
invoke mux 1
Actual useful processing performed by the modules
Return to dispatch code

FIG. 20

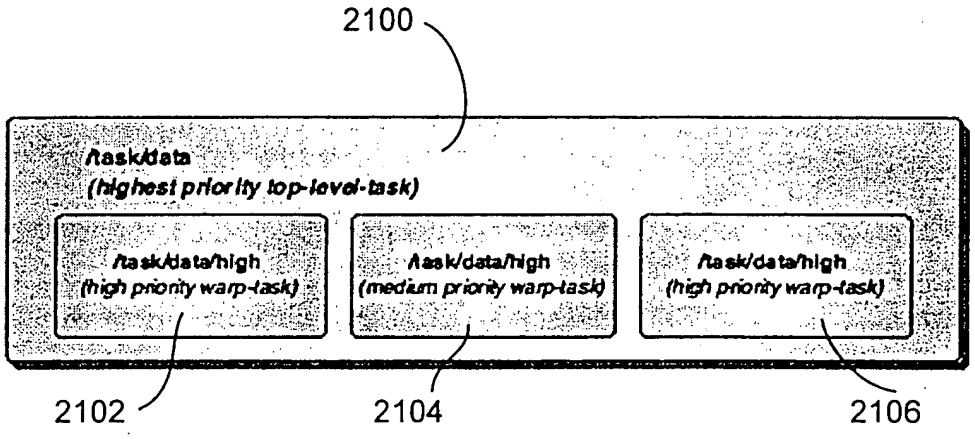
2100
/task/data
(highest priority top-level-task)
/task/data/high
(high priority warp-task)
/task/data/high
(medium priority warp-task)
/task/data/high
(high priority warp-task)
2102
2104
2106

FIG. 21